# Lessons Learned from Service-Oriented Systems for Engineering Systems of Systems

Software Engineering Institute
Carnegie Mellon University

Grace A. Lewis (glewis@sei.cmu.edu)
SATURN 2010
May 20, 2010

Software Engineering Institute | Carnegie Mellon

© 2010 Carnegie Mellon University

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **20 MAY 2010** | | **00-00-2010 to 00-00-2010** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Lessons Learned from Service-Oriented Systems for Engineering Systems of Systems** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

**There is an increasing trend towards interconnected systems of systems (SoSs) that provide capabilities that are not available in a single system. Many organizations, including the DoD, are already implementing these SoSs. However, existing software and system engineering practices do not scale well to SoS. SoS engineering is still an open problem with significant challenges. Understanding these challenges and providing engineering solutions will require a two-pronged approach ? A top-down approach that deals with understanding SoS at an abstract level. This view is essential to understand the key concerns that exist in any SoS independent of the concrete technologies that are used to implement the SoS. ? A bottom-up approach that focuses on abstracting the concepts and lessons learned from specific examples of engineering SoSs. Currently, the most common approaches for engineering software-intensive SoSs are service-oriented architecture (SOA), grid computing, and cloud computing, all of which are distributed computing paradigms. In the future, newer technologies may replace or complement these existing engineering approaches. This presentation focuses on the bottom-up approach by exploring areas where lessons learned from implementation of service-oriented systems are abstracted and applied to SoS.**

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **31** | |

# Agenda

**Service Orientation and Systems of Systems** ⇦

What Has Worked Well in SOA Implementations

What Does Not Yet Work Well in SOA Implementations

Final Thoughts


© 2010 Carnegie Mellon University

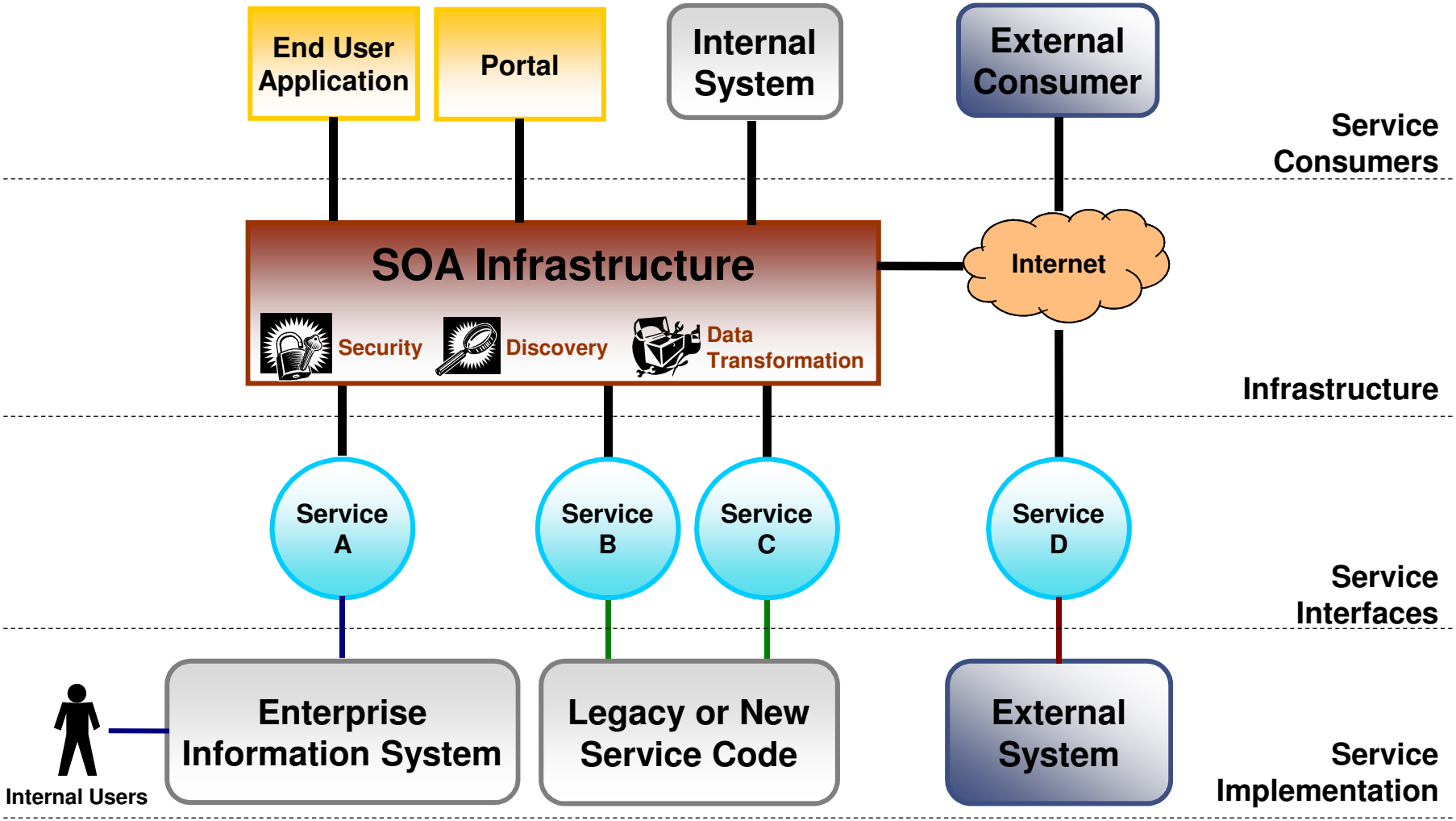
# Service Orientation

Service orientation has become a common approach for implementation of distributed, loosely-coupled systems

- Services provide reusable business functionality via well-defined interfaces.
- Service consumers are built using functionality from available services.
- There is a clear separation between service interface and service implementation.
  - Service interface is just as important as service implementation.
- An SOA infrastructure enables discovery, composition, and invocation of services.
- Protocols are predominantly, but not exclusively, message-based document exchanges.


© 2010 Carnegie Mellon University

# Components of a Service-Oriented System





© 2010 Carnegie Mellon University

# Benefits Associated with Service Orientation

Cost-Efficiency

- Services provide functionality that can be reused many times by many consumers
- Services become a single point of maintenance and management for common functionality

Agility

- Via service discovery mechanisms, developers can find and take advantage of existing services to reduce development times

Legacy Leverage

- Separation of service interface from service implementation provides true platform independence

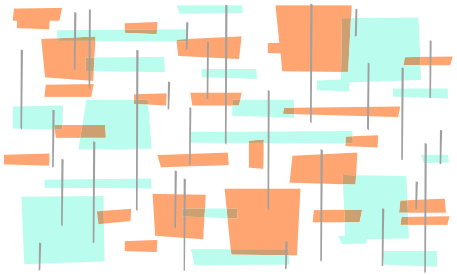
Adaptability

- Separation of service interface from service implementation allows for incremental deployment of services and incremental modernization


© 2010 Carnegie Mellon University

# System of Systems (SoS)

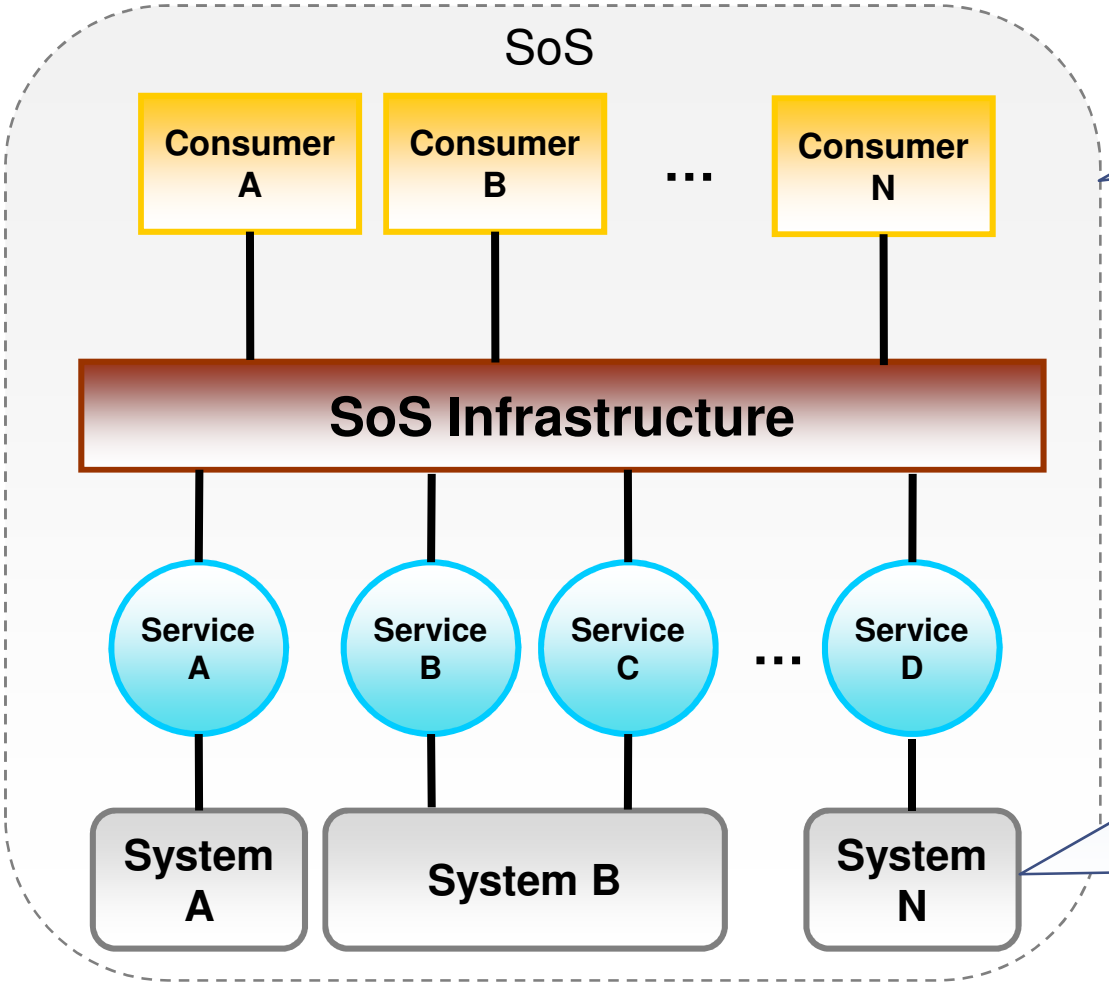
A System of Systems is "*a set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities*."*



* OSD Systems Engineering Guide


© 2010 Carnegie Mellon University

# Notional View of a Service-Oriented SoS




Characteristics of SoS*

- Evolutionary development
- Emerging behavior

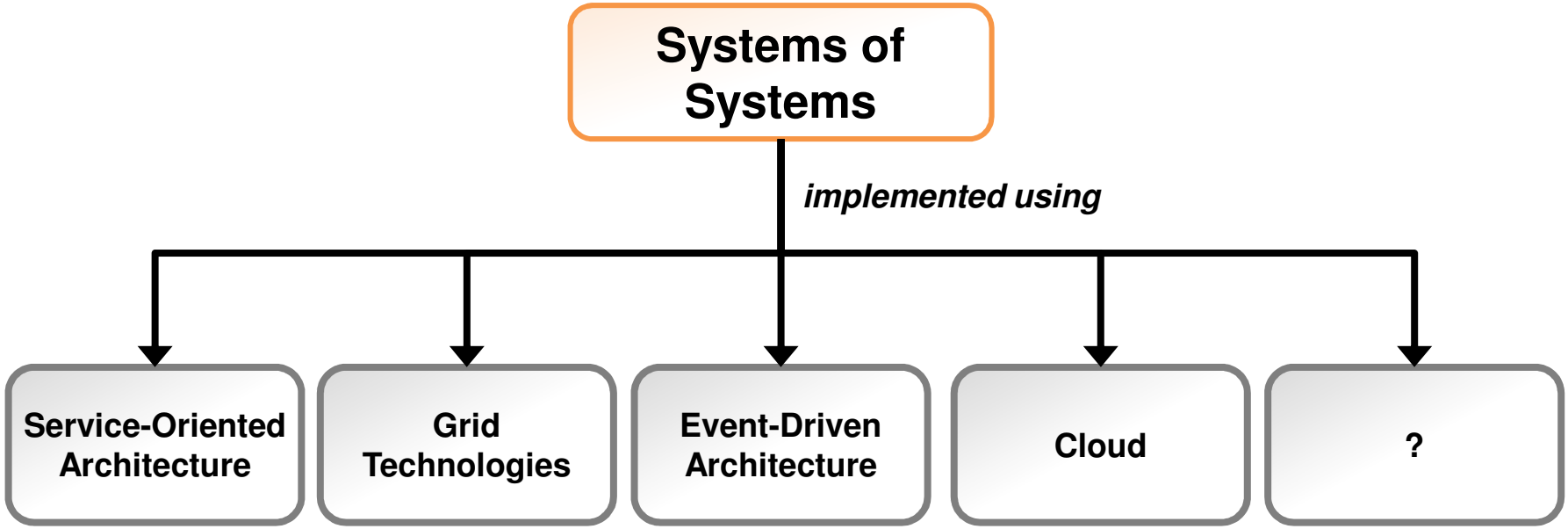
Characteristics of Constituent Systems*

- Operational independence
- Managerial independence
- Geographic distribution

* Maier, M. (1998), "Architecting Principles for Systems-of-Systems," Systems Engineering, Vol. 1, No. 4 (pp. 267-284)


© 2010 Carnegie Mellon University

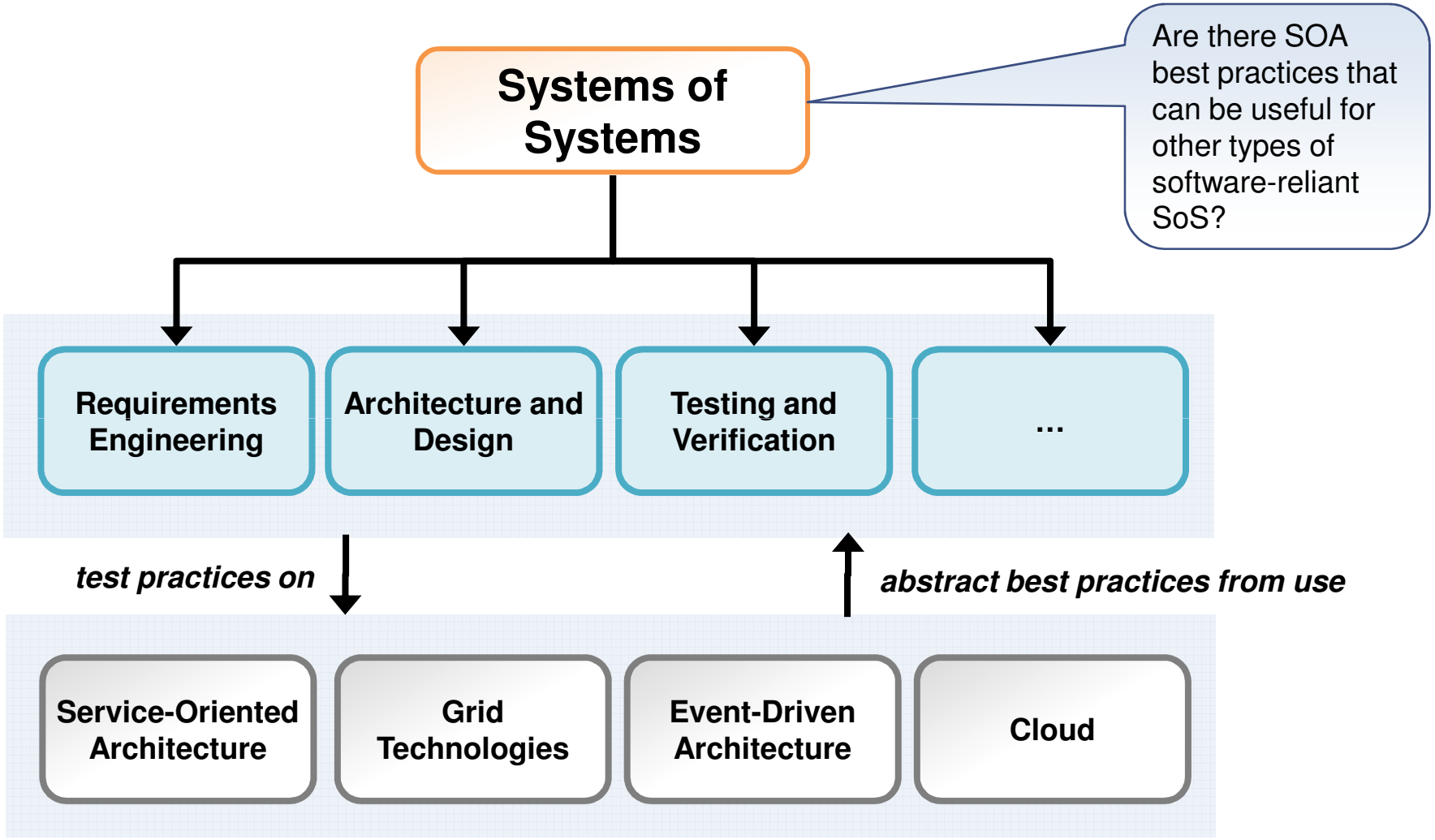
# SOA is One Implementation Technology for SoS





© 2010 Carnegie Mellon University

# What Can We Learn from SOA Implementations?





© 2010 Carnegie Mellon University

# Agenda

Service Orientation and Systems of Systems

**What Has Worked Well in SOA Implementations** 

What Does Not Yet Work Well in SOA Implementations

Final Thoughts


© 2010 Carnegie Mellon University

# What Are Some Things That Have Worked Well in SOA Implementations?

Standardization

Loose coupling

Strategic service identification

Service discovery mechanisms

Governance

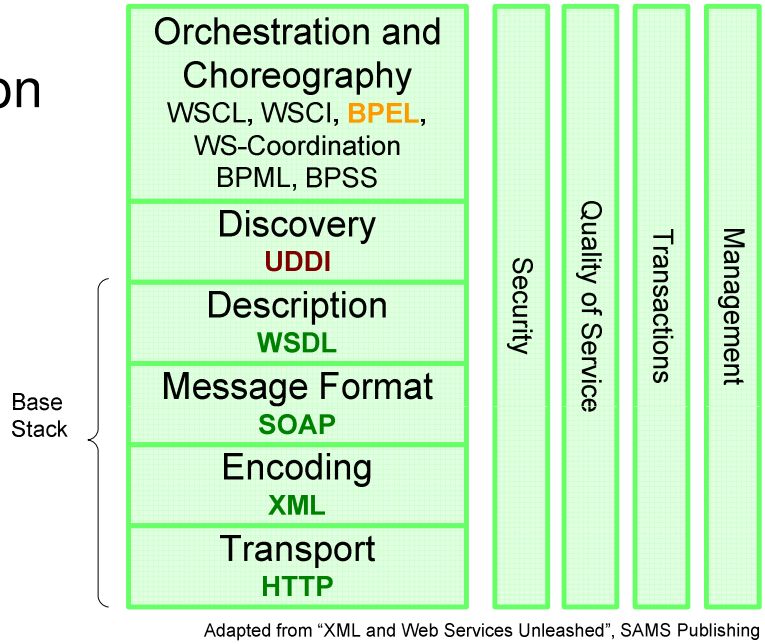

© 2010 Carnegie Mellon University

# Standardization

Even though there are multiple ways to implement service-oriented systems, the most common implementation is based on WS* Web Services

Benefits

- Interoperability: standard interfaces to heterogeneous technologies
- Tool support
- Enablement of other aspects of service-oriented systems: discovery, composition, etc
- Usage of third-party services
- Potential for shorter development times
  - All you need to know to use a service is there




Adapted from "XML and Web Services Unleashed", SAMS Publishing


© 2010 Carnegie Mellon University

# Standardization: How Does this Bridge to SoS?

Bottom Line: Some level of standardization is necessary

- A very controversial topic in many SoS settings

Benefits for SoS

- Encapsulation of complexity and heterogeneity of constituent systems
  - Operational independence
  - Managerial independence
  - Geographic distribution
- Greater tool support
- Enablement of interface-based testing


© 2010 Carnegie Mellon University

# Strategic Service Identification

Service identification is done in the problem domain

- Based on the premise that what changes is business and not the technology
- Starts from business goals for SOA adoption
  - Top-Down: Identification of business processes that support business goals
  - Bottom-Up: Identification of legacy capabilities that support business goals

Services represent reusable business capabilities

- Leads to greater and easier reuse

Target of service identification techniques is low coupling and high cohesion of service capabilities

- Also related to stateless nature of services


© 2010 Carnegie Mellon University

# Strategic Service Identification: How Does This Bridge to SoS?

What is done to identify services can be done to identify capabilities

- Business processes map to SoS usage scenarios
- Services map to individual system capabilities

Advantage is that individual systems already have high cohesion and low coupling

- At least from a consumer perspective


© 2010 Carnegie Mellon University

# Loose Coupling

Different architectural patterns emphasize different forms of loose coupling — however, there is always some form of coupling

- Data centric — data model
- Event driven — events and event mechanism
- Service orientation — interfaces and communication mechanisms

Two forms of loose coupling in service orientation

- Between provider and consumer
  - Service provider and consumer know as little as possible about each other
  - SOA infrastructure mediates a lot of the differences between providers and consumers
  - SOA infrastructure also provides centralization of support for key quality attributes
- Between interface and implementation
  - Service location
  - Service implementation technologies


© 2010 Carnegie Mellon University

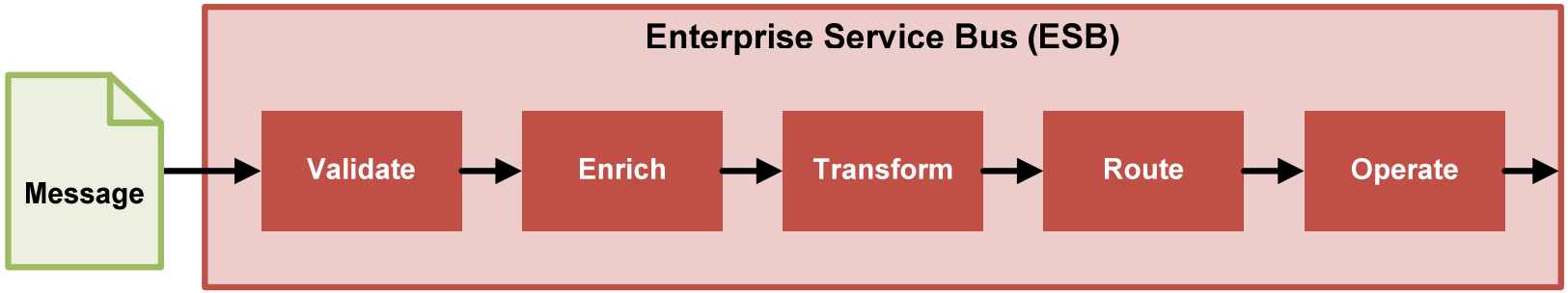
# Loose Coupling: How Does this Bridge to SoS?

Hide technology of constituent systems

- Technology should not bleed through system interfaces, e.g. obfuscation of error messages

Provide integration mechanisms that are technology-neutral, e.g. XML

Provide an infrastructure to mediate differences between constituent systems, e.g. the ESB VETRO pattern




Source: Dave Chappell, "Enterprise Service Bus" (O'Reilly: June 2004, ISBN 0-596-00675-6)


© 2010 Carnegie Mellon University

# Service Discovery

In a service-oriented environment, as services are created they are published in a place that is accessible to consumers and can be queried for desired capabilities, e.g. service registry, web page, directory

Some environments have service repositories with additional service metadata that can be used to identify desired capabilities

- Specification (Contract) — minimum required information
- Description
- Classification
- Usage history
- Test case
- Test results
- Quality metrics
- Documentation


© 2010 Carnegie Mellon University

# Service Discovery: How Does This Bridge to SoS?

Consider having a capability repository

- There is considerable tool support that can be leveraged
- Include in it everything that is needed to reuse a capability — minimizes interaction between developers and therefore promotes agility

Metadata should be published along with the capability, not only to help with the discovery process, but also to know how to use that capability once it is discovered

- Quality attributes, e.g., performance limitations
- Assumptions, e.g., encryption mechanisms
- Constraints, e.g., usage context

Keep in mind that system-capabilities-as-a-service is another “quantum leap” in granularity


© 2010 Carnegie Mellon University

# Governance

SOA Governance is the set of policies, rules, and enforcement mechanisms for developing, using, and evolving service-oriented systems, and for analysis of the business value of those systems

- Design-time governance
- Runtime governance
- Change-time governance

SOA Governance was created to solve a problem created by SOA

- Way to add control and enforce consistency

Many (but not all) aspects of SOA governance can be automated and embedded in the SOA infrastructure, e.g.

- Execution of services according to policies
- Runtime logging and monitoring


© 2010 Carnegie Mellon University

# Governance: How Does This Bridge to SoS?

Something similar to SOA Governance is necessary to provide agreement on aspects such as

- Characteristics of central registry
- Design-time, runtime and change-time policies
- Service Level Agreements (SLAs)
- Tool support

Due to the independent nature of constituent systems, change-time governance is key to SoS

- How are problems in constituent systems reported?
- What happens if a constituent system changes?
- How are changes and upgrades in constituent systems communicated?
- How will capabilities of constituent services be tested?


© 2010 Carnegie Mellon University

# Agenda

Service Orientation and Systems of Systems

What Has Worked Well in SOA Implementations

**What Does Not Yet Work Well in SOA Implementations** 

Final Thoughts


© 2010 Carnegie Mellon University

# Things That Have Not Been Solved in the SOA World 1

Multi-organizational SOA implementations, e.g.

- Distributed development tasks, e.g. assurance
- Multi-organizational concerns, e.g. trust, federation, security

Standardization on how to specify quality attributes

- Key for service discovery and SLA management and monitoring

Support for interoperability at higher levels

- Above syntactic: semantic and process interoperability

Automation of service discovery

- Many efforts, but still not widely used


© 2010 Carnegie Mellon University

# Things That Have Not Been Solved in the SOA World 2

Many aspects of SOA governance cannot be automated

- As stated by AgilePath: “Governance is behavior (90%), not technology (10%)”
- Makes it difficult to enforce

Until these things are solved, it will be difficult to build virtual SoS where capabilities are identified and invoked on the fly


© 2010 Carnegie Mellon University

# Agenda

Service Orientation and Systems of Systems

What Has Worked Well in SOA Implementations

What Does Not Yet Work Well in SOA Implementations

**Final Thoughts** ⇦


© 2010 Carnegie Mellon University

# SoS Types 1

Maier defines four types of SoS based on their management structure*

- Directed: the systems are integrated and built to fulfill specific purposes
- Acknowledged: SoS has recognized objectives, a designated manager, and resources
- Collaborative: constituents voluntarily agree to fulfill central purposes
- Virtual: no central authority or centrally agreed purpose

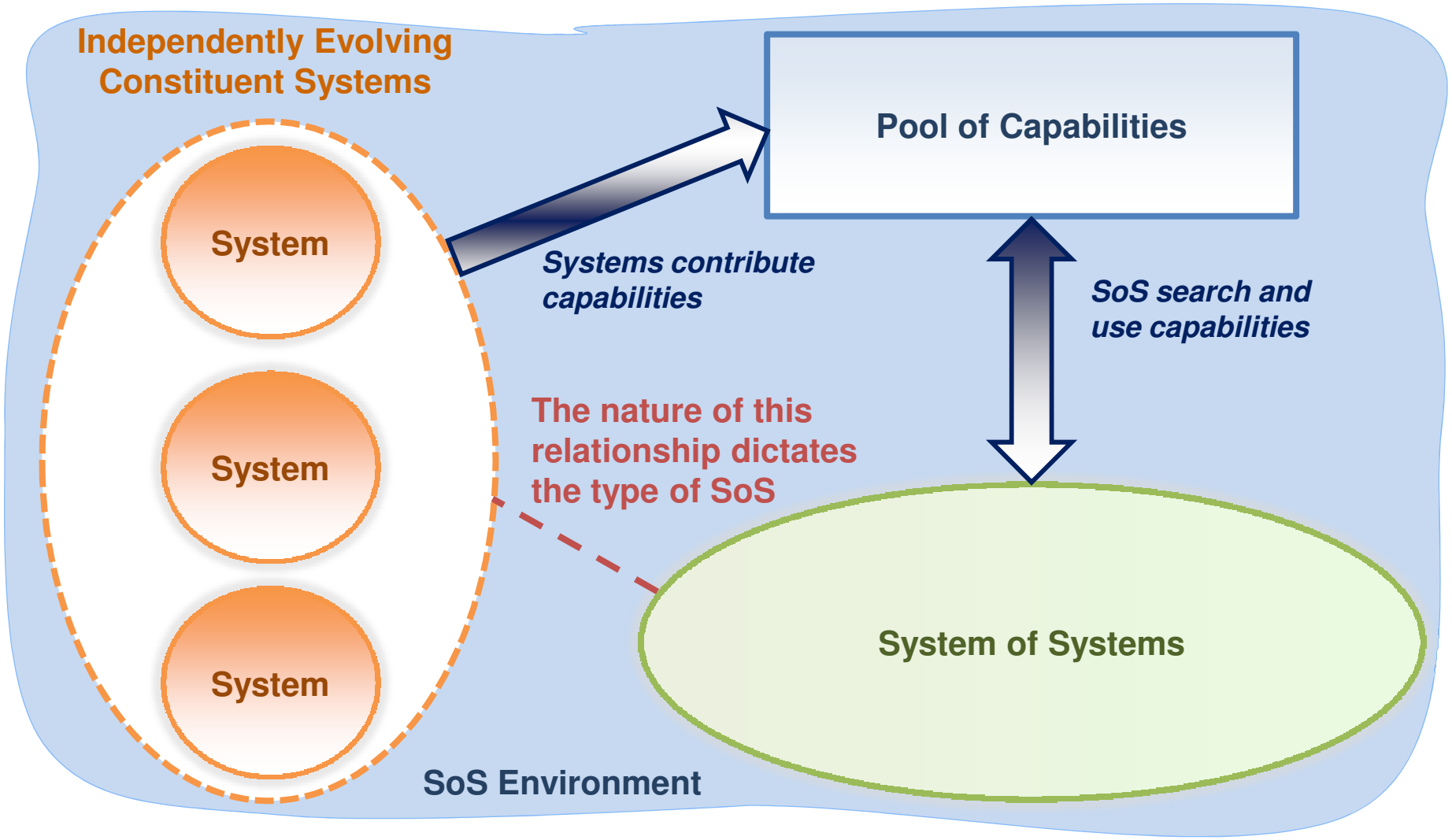
Current technologies used in service-oriented systems can support directed and acknowledged SoS

- Requirements of collaborative and virtual SoS are beyond technology

* Maier, M. (1998), "Architecting Principles for Systems-of-Systems," Systems Engineering, Vol. 1, No. 4 (pp. 267-284)


© 2010 Carnegie Mellon University

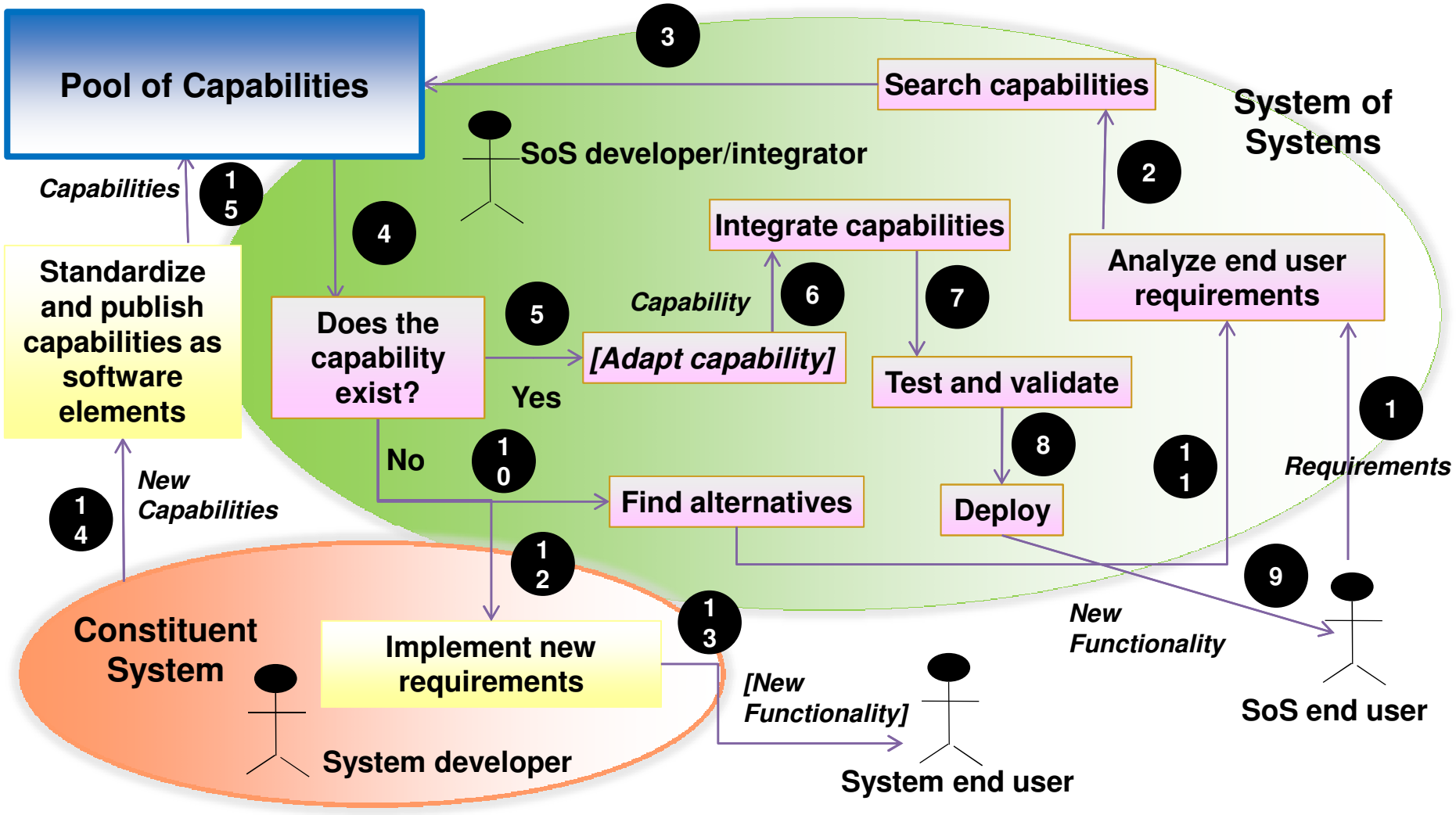
# SoS Types 2





© 2010 Carnegie Mellon University

# The Idealized Service-Oriented SoS Process





© 2010 Carnegie Mellon University

# Need to Separate Service-Orientation from SOA Implementation Technologies

The concept of service-orientation is here to stay, but the technologies will change over time to meet new requirements

- It is important to separate service-orientation as a concept from SOA as a set of technologies to support service-orientation

A challenge for SoS architects is to reduce the impact of changing technologies from the implementation of service-oriented concepts

- "Separation of concerns on steroids"

Developers of SoS will have to accept the lack of control over SoS elements, independent of the implementation technology, e.g.

- For assurance, rely on monitoring in addition to testing
- Use defensive programming, e.g. exception handling, fallback strategies


© 2010 Carnegie Mellon University

# Questions?


© 2010 Carnegie Mellon University

**NO WARRANTY**

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.


© 2010 Carnegie Mellon University